Filling Out the Table Binary Data Structure

The ***<Table_Binary>*** class contains the data structure definition for a binary table. Each row in the table has the same structure, defined in a *Record_Binary* class. Records are themselves composed of scalar fields, or sub-records called "grouped fields". Use it if your data cannot be reasonably archived as fixed-width character tables because of data volume.

For additional explanation, see the PDS4 *Standards Reference*, or contact your PDS node consultant.

Following are the attributes and subclasses you will find in <*Table_Binary*>, in label order.

Note that in the PDS4 master schema, all classes have capitalized names; attributes never do.

<name>

OPTIONAL

If you'd like to give this table a descriptive name, here's the place to do it.

<local_identifier>

OPTIONAL

If you want to reference this *Table_Binary* from somewhere else in this label, give it a formal label here. Use an identifier that would make a valid variable name in a typical programming language, and you should be OK syntactically.

<md5_checksum>

OPTIONAL

Use this attribute to provide the MD5 checksum of the object *only*. If the object occupies the entire file, then the checksum should be given as an attribute of the <*File*> object. This checksum should be calculated using *only* the bytes defined as being part of this table.

<offset>

REQUIRED

The offset, in bytes, from the beginning of the file holding the table data to the first byte in the table. You must specify the unit for this keyword, like this:

```
    <offset unit="byte">0</offset>
```

<records>

REQUIRED

This is the total number of records in the table. Records in binary tables do not usually have record delimiters. On the rare occasions when they do, they are documented as a just another field in the record.

<description>

OPTIONAL

This attribute provides a place for additional free-format text comments.

<Uniformly_Sampled>

OPTIONAL

If this *Table_Binary* contains records which are uniformly spaced in some dimension (time, wavelength, distance, etc.), you can use this class to define that dimension and interval rather than including an additional field in each row to hold the value explicitly. The details are on the Filling Out the Uniformly Sampled Class page.

<Record_Binary>

REQUIRED

This class describes the structure of one complete record in the table.

<fields>

REQUIRED

The number of *Field_Binary* classes directly under (that is, in the first nesting level of) the *Record_Binary* class. Do ***not*** count *Field_Binary* classes nested under *Group_Field_Binary* classes.

If your *Record_Binary* contains only one or more *Group_Field_Binary* classes, this will have a value of zero.

<groups>

REQUIRED

The number of *Group_Field_Binary* classes directly under (that is, in the first nesting level of) the *Record_Binary* class. Do ***not*** count *Group_Field_Binary* classes under other, nested *Group_Field_Binary* classes.

If your *Record_Binary* contains only one or more *Field_Binary* classes, this will have a value of zero.

<record_length>

REQUIRED

The total length of the record, including all fields, all repetitions of group fields, and any fill space between fields. You must specify a unit of bytes for this value:

```
    <record_length unit="byte">1234</record_length>
```

A Note about Fields and Group_Fields

Binary table records are composed of *Field_Binary* and *Group_Field_Binary* classes. A *Record_Binary* must have at least one of those (either will do), and can have an arbitrary number of them, in any order (that is, you can have *Field_Binary* and *Group_Field_Binary* classes interspersed). Note, however, that *Group_Field_Binary* classes are ***never*** necessary - they are a notational convenience to save writing out large numbers of essentially identical *Field_Binary* definitions.

<Field_Binary>

OPTIONAL

The class defines a single scalar field.

<name>
REQUIRED

The name of the field. SBN recommends that this be something fairly human-readable that can be easily turned into a variable name for use in applications, or displayed as a meaningful column heading.

<field_number>
OPTIONAL

This is the sequential number of the *Field_Binary* definition. The *field_number* is intended to be a help to human readers trying to map field definitions to columns in a print-out of the *Table_Binary*.

> The *Standards Reference* lays out rules for using the *field_number* in cases where there are *Group_Field_Binary* classes present which can be useful in programmatic contexts, but not so much in the visual-inspection case.

<field_location>
REQUIRED

This is the location, in bytes, of the first byte in the field relative to the immediately enclosing structure. (Note that locations begin with one, rather than zero.) So if this field is at the first level under a <Record_Binary> class, the location is relative to the first byte of the record. If the field is inside a <Group_Field_Binary>, the location is relative to the first byte of that group.

You must indicate a unit of bytes for this field:

```
<field_location unit="byte">1</offset>
```

<data_type>
REQUIRED

The type of the values in the field. This must be one of the values listed in the Standard Values Quick Reference for binary table fields.

<field_length>
REQUIRED

The length of the field, in bytes. You must specify the unit:

```
<field_length unit="byte">12</field_length>
```

<field_format>
OPTIONAL

The value of this attribute is a string representing the optimal print format for the data in the field, using a subset of the POSIX print conventions defined in the *Standards Reference*, and also described on the PDS4 Field Format Conventions page.

<unit>
OPTIONAL

If the value in this field has an associated unit, this is where it goes. This value is case sensitive, and you may use characters from the UTF-8 character set (like the Angstrom symbol) where appropriate.

Note: If a field contains a unitless value, then there should be no *<unit>* attribute. **NEVER** include a null *unit* value, or even worse, this: `<unit>N/A</unit>`.

<scaling_factor>
OPTIONAL

If the observational values of this field were scaled prior to writing, this attribute should contain the value by which the data must be multiplied in order to get back to the original value. Scaling factors are applied prior to adding any offset.

<value_offset>
OPTIONAL

If the observational values of the field were shifted by an offset prior to writing, this attribute should contain the value that must be added to each field value to get back to the original value. Offsets are added after the scaling factor, if any, is applied.

<description>
OPTIONAL

Free-format text describing the content of the field.

<Special_Constants>
OPTIONAL

This class defines flag values used to indicate that a particular field value is unknown for one reason or another. It is identical to the *<Special_Constants>* class used in the *Array* classes. For details, check the Filling Out the Array 2D Data Structure - <Special_Constants> page. Here is a quick list of the special constants available in this class:

- *saturated_constant*
- *missing_constant*
- *error_constant*
- *invalid_constant*
- *unknown_constant*
- *not_applicable_constant*
- *valid_maximum*
- *high_instrument_saturation*
- *high_representation_saturation*
- *valid_minimum*
- *low_instrument_saturation*
- *low_representation_saturation*

<Field_Statistics>
OPTIONAL

If you want to include things like extrema, mean value, and such for all the values that occur in this field through all the records in the table, this is the place to do it. This class is identical for all *Field* types. For details, see Filling Out the Field Statistics Class. Here is a quick list of the field statistics available in this class:

- *local_identifier*
- *maximum*
- *minimum*
- *mean*
- *standard_deviation*

- *median*
- *description*

<Packed_Data_Fields>
OPTIONAL

When several values are packed into a single field, ignoring byte and word boundaries, the set of values is first defined as a single <*Field_Binary*>, which of course has an integral number of bytes. Then, the individual values that were packed into the string of bytes are defined as a series of <*Field_Bit*> classes inside this *Packed_Data_Fields* class.

<Group_Field_Binary>

OPTIONAL

This class defines a set of *Field_Binary* and/or *Group_Field_Binary* classes that repeats a given number of times in each record. *Group_Field_Binary* classes may be nested.

<name>
OPTIONAL

If you'd like to give your group a name, which can be useful for display programs and reviewers wondering why a group exists, you can do it here.

<group_number>
OPTIONAL

Analogous to *field_number* for scalar fields, this is a sequential number useful for referencing *Group_Field_Binary* classes at a single nesting level of a complex *Record_Binary* definition.

<repetitions>
REQUIRED

The number of times the complete set of *Field_Binary* and nested *Group_Field_Binary* classes comprising the current <*Group_Field_Binary*> repeats.

<fields>
REQUIRED

The count of *Field_Binary* classes directly under (i.e., at the first nesting level below) the *Group_Field_Binary* definition. This will be zero if the group contains no scalar fields.

<groups>
REQUIRED

The count of *Group_Field_Binary* classes directly under (i.e., at the first nesting level below) the present *Group_Field_Binary* definition. This will be zero if the group contains no nested *Group_Field_Binary* classes.

<description>
OPTIONAL

This attribute is for any additional description or explanation you might like to provide.

<group_location>
REQUIRED

This is the location of the first byte of the first field of the first repetition of this group relative to the containing *Record_Binary* or *Group_Field_Binary* location. If the group starts at the beginning of the containing *Record_Binary* or *Group_Field_Binary*, this has a value of one. You must specify a unit of "byte" for this value. For example:

```
    <group_location unit="byte">1</group_location>
```

This value should be set bearing in mind that it will be used as the base offset for locating each repetition of the group. In other words, the location of the start of the first repetition of the group is *group_location* + 0*(*group_length*/*repetitions*); the location of the start of the second repetition of the group is *group_location* + 1*(*group_length*/*repetitions*), and so on.

<group_length>
REQUIRED

This is the total length of the group within the record. That is, ***it is the total length of all the fields and nested groups, multiplied by the value of the* <repetitions> *attribute***, above. It must be in bytes, and you must include a *unit* of "bytes" in the attribute, as for *group_location*. It must be evenly divisible by the value of *repetitions*.

Note: *This value is **not** validated. Calculate carefully.*

<Fields> and Nested <Group_Fields>

As in the *Record_Binary*, the *Group_Field_Binary* may contain either *Field_Binary* classes, *Group_Field_Binary* classes, or both intermixed. *Group_Field_Binary* classes may be nested arbitrarily deeply. The requirement for these data structure classes inside a <*Group_Field_Binary*> are identical to those above for inside a *Record_Binary*.